CLAIMS

Having thus described our invention, what we claim as new and desire to secure by Letters Patent is as follows:

1. A method of compressing an extensible markup language (XML) document, comprising:

compressing an XML document and its associated schema information such that information in a markup portion therein is maintained in a compressed form to allow the document to be reconstructed,

wherein, during said compressing, said markup portion and a non-markup portion of said document are separated, and the non-markup component is compressed using a first compression method and the markup component is compressed using a second compression method.

2. The method according to claim 1, wherein said mark-up portion comprises structured component information and wherein said schema information associated with the document is used with compressing the structure component to obtain a predetermined higher compression rate while simultaneously retaining the structure.

3. The method according to claim 2, wherein said schema information comprises a document type definition (DTD).

4. The method according to claim 1, wherein said markup portion comprises a structure of the document and said non-markup portion comprises data associated with said document.

5. The method according to claim 1, wherein said XML document includes elements selectively containing any of character data, child elements, or a combination of character data and child elements.

6. The method according to claim 1, wherein an XML markup language is defined in a Document Type Definition (DTD),

wherein said DTD is contained in at least one of a <!DOCTYPE> tag and an external file and referenced from a <!DOCTYPE> tag.

7. The method according to claim 1, wherein an element is defined as a group of one or more subelements, character data, EMPTY, or ANY, and

wherein attributes are optional, required, or selectively have a fixed value, and wherein optional attributes have a default and fixed attributes always have a default.

8. The method according to claim 6, wherein prior to compressing, a tag name token is associated with each tag name, an attribute name token is associated with each attribute name and an attribute value token is associated with each attribute value which is defined in the DTD.

9. The method according to claim 8, wherein a server searches for the token corresponding to an element or an attribute.

10. The method according to claim 9, wherein a client searches for the element or attribute corresponding to a particular token.

11. The method according to claim 10, wherein the server and the client each use a different data structure.

12. The method according to claim 1, wherein said compressing further comprises:

creating the data structure on a server and on a client.

13. The method according to claim 12, wherein said compressing further comprises:

parsing a document type definition (DTD) of the XML document.

14. The method according to claim 13, wherein said compressing further comprises:

filling in server code spaces.

15. The method according to claim 14, wherein said compressing further comprises:

filling in client code spaces.

16. The method according to claim 14, wherein, for a specific tag name or attribute name or value, the server finds a corresponding token.

17. The method according to claim 16, wherein a hash table is used for storing a key and a value as a couple where the key is a string and the value is a corresponding token.

18. The method according to claim 17, wherein the client finds the tag name or attribute name or value corresponding to a specific token, and wherein said token is split into a set of overlapping code spaces.

19. The method according to claim 18, wherein said client is provided with a page number and an index in a code space page, and finds the corresponding string, and wherein each code space includes a two-dimensional array indexed by page numbers and indexes in pages.

20. The method according to claim 14, wherein the DTD is parsed using a document object model (DOM) parser which generates a DOM structure for the DTD, for allowing accessing elements declarations and attributes declarations with name and type and attributes values when an attribute type is enumerated.

21. The method according to claim 14, wherein server code spaces are filled-in by filling in a hash table for an element code space such that a page number variable is set to a first predetermined value and the index variable is set to a second predetermined value.

22. The method according to claim 21, wherein, for each element declaration, an element name is added to the hash table with the element name as the key and (256 x pageNumber + index) as the value,

wherein the index is incremented by 1 and a size of a page for elements is a predetermined number and when the index reaches the predetermined number, the page number is incremented by 1 and the index is reset, and

wherein when the page number reaches a maximum value, an exception is raised.

23. The method according to claim 22, wherein for the attribute code space, for each element declared, a corresponding attribute declaration is obtained from a previously built Document Object Model (DOM) tree, and adds the attribute name in the hash table with the attribute name as the key and (256 x pageNumber + index) as the value.

24. The method according to claim 23, wherein, if the attribute type is enumerated, then the values of the enumerated attribute are searched,

for each value, the attribute value is added in the hash table with the attribute value as the key and (256 x pageNumber + index) as the value, and the index for the value is incremented,

wherein a size of a page for an attribute value is a predetermined number such that when the index reaches the predetermined number, the system increments the page number and resets the index, and

wherein, when the page number reaches its maximum value, an exception is raised,

wherein, if there are no values or when the values have been successfully added to the attribute value code space, then the index is incremented for the name,

wherein the size of a page for an attribute name is a predetermined number and such that when the index reaches the page number is incremented and the index is reset, and when the page number reaches its maximum value, an exception is raised.

25. The method according to claim 22, wherein the filling-in of the client code spaces includes:

setting the page number variable to a first predetermined number and the index variable to a second predetermined number,

for each element declaration, obtaining the element name, adding it in the elements array at a predetermined position indicating (page number, index) and incrementing the index.

26. The method according to claim 25, wherein for an attribute code space, for each element declared, obtaining the corresponding attribute declaration from a previously built DOM structure, adding the attribute name in the attribute names array at a predetermined position represented by a couple formed of (page number, index).

27. The method according to claim 26, wherein if the attribute type is enumerated, then checking for the values of this enumerated attribute, and

for each value, adding the attribute value in the attribute values array at a predetermined position (page number, index) and incrementing the index for the value,

wherein when the index reaches a predetermined value, incrementing the page number and resetting the index to a predetermined number, and ,

when the page number reaches a maximum value, raising an exception.

28. The method according to claim 27, wherein if there are no values or when the values have been added to the attribute value code space, incrementing the index for the name, and

wherein when the size of a page for attribute name is a predetermined number, incrementing the page number to a predetermined number and resetting the index to a predetermined number, and

when the page number reaches a maximum value, raising an exception.

29. The method according to claim 1, further comprising:

merging an attribute names code space and an attribute values code space into one token.

30. The method according to claim 1, wherein each couple formed by (attribute name, attribute value) is made into a single token.

31. The method according to claim 30, wherein, for each element declared, a corresponding attribute declaration is obtained, and wherein if an attribute-type is not enumerated by having a specific value declared for this attribute, then adding the attribute name in the attribute code space.

32. The method according to claim 31, wherein the attribute code space comprises a hash table for a server, and an array for a client.

33. The method according to claim 31, wherein if the attribute-type is enumerated, then values are searched for the enumerated attribute, and for each value, the couple (attribute name, attribute value) is added with a specific token in the attribute code space.

34. The method according to claim 33, wherein, when the server encounters an attribute with a value, the server searches the attribute code space for the couple (attribute name, attribute value), and

if the server finds the couple, the server sends the token associated therewith.

35. The method according to claim 34, wherein if the server cannot find the couple, the server looks for the attribute name in the attribute code space, and

wherein, if the name is found, the server sends the corresponding token for the name followed by a string inline token followed by the attribute value encoded in a charset specified at a beginning of the XML document stream.

36. The method according to claim 21, wherein the element code space includes elements selectively having required or fixed attributes, and wherein, for said elements having the required or fixed attributes tokens are not transmitted,

wherein the names of the required or fixed attributes with the element name are stored in the element code space.

37. The method according to claim 36, wherein to fill in the element code space, for each element declaration, the element name and required and fixed attributes are obtained, and

wherein the element names and the required and fixed attribute names are added to the element code space, and for the fixed attributes, a value thereof is added additionally.

38. The method according to claim 37, wherein, in the attribute code space, only implied attributes are stored with their value if defined.

39. The method according to claim 1, wherein said first compression method comprises a lossless data compression method.

40. The method according to claim 39, wherein said second compression method comprises a binary encoding method.

41. A method of compressing an extensible mark-up language (XML) document, comprising:

creating a data structure on a server and on a client;

parsing a document type definition (DTD) of the XML document;

filling in the server code spaces; and

filling in the client code spaces.

42. A system for compressing an extensible markup language (XML) document, comprising:

means for compressing an XML document such that information comprising a markup portion therein is maintained in a compressed form to allow the document to be reconstructed; and

means, during compressing, for separating said markup portion and a non-markup portion of said document, and wherein said compressing means compresses the non-markup component using a first compression method and compresses the markup component using a second compression method.

43. A programmable storage medium tangibly embodying a program of machine-readable instructions executable by a digital processing apparatus to perform compressing of an extensible markup language (XML) document, said method comprising:

compressing an XML document such that information comprising a markup portion therein is maintained in a compressed form to allow the document to be reconstructed,

wherein, during said compressing, said markup portion and a non-markup portion of said document are separated, and the non-markup component is compressed using a first compression method and the markup component is compressed using a second compression method.

44. A programmable storage medium tangibly embodying a program of machine-readable instructions executable by a digital processing apparatus to perform compressing of an extensible markup language (XML) document, said method comprising:

creating a data structure on a server and on a client;

parsing a document type definition (DTD) of the XML document;

filling in the server code spaces; and

filling in the client code spaces.

5